



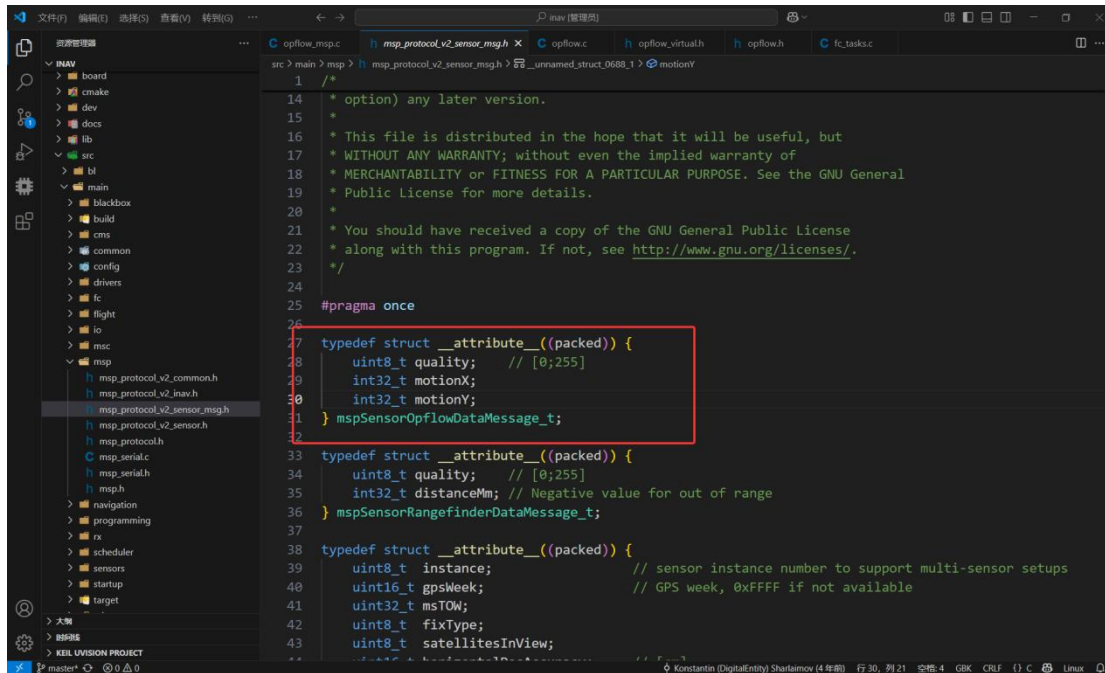
优象光流+激光测距模块接入 INAV 开源飞控的使用教程

目 录

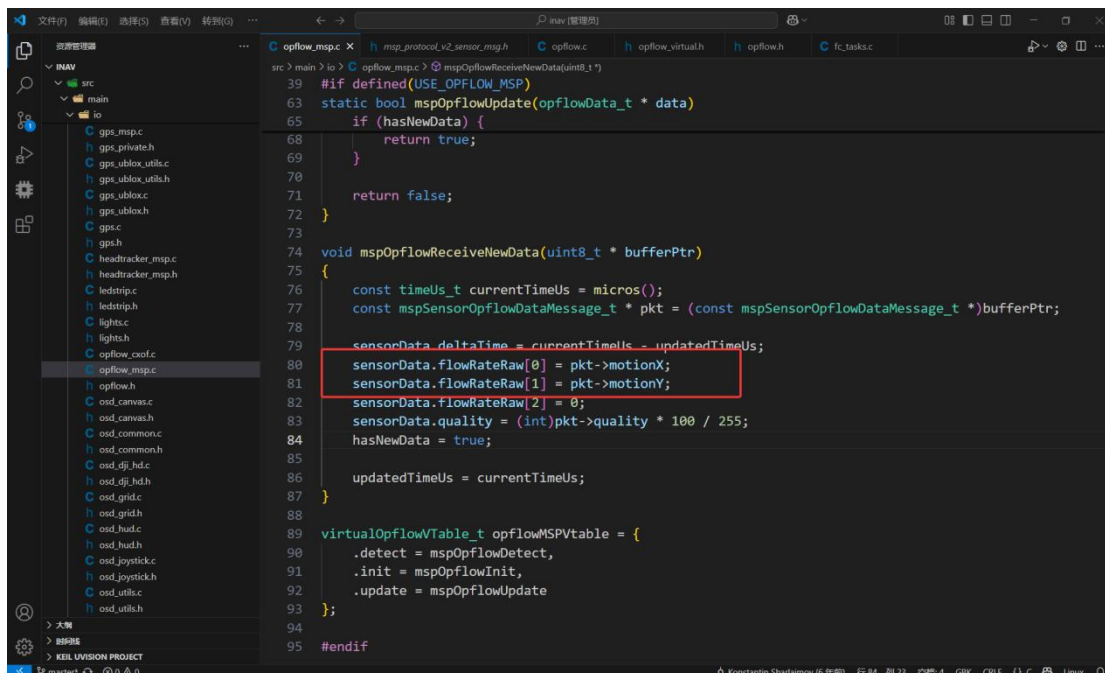
1 INAV 固件修改方法	1
2 INAV 固件编译	2
2.1 环境安装	2
2.2 INAV 源代码的拉取和编译	3
3 INAV 固件烧录	5
4 INVA 地面站的使用	6

1 INAV 固件修改方法

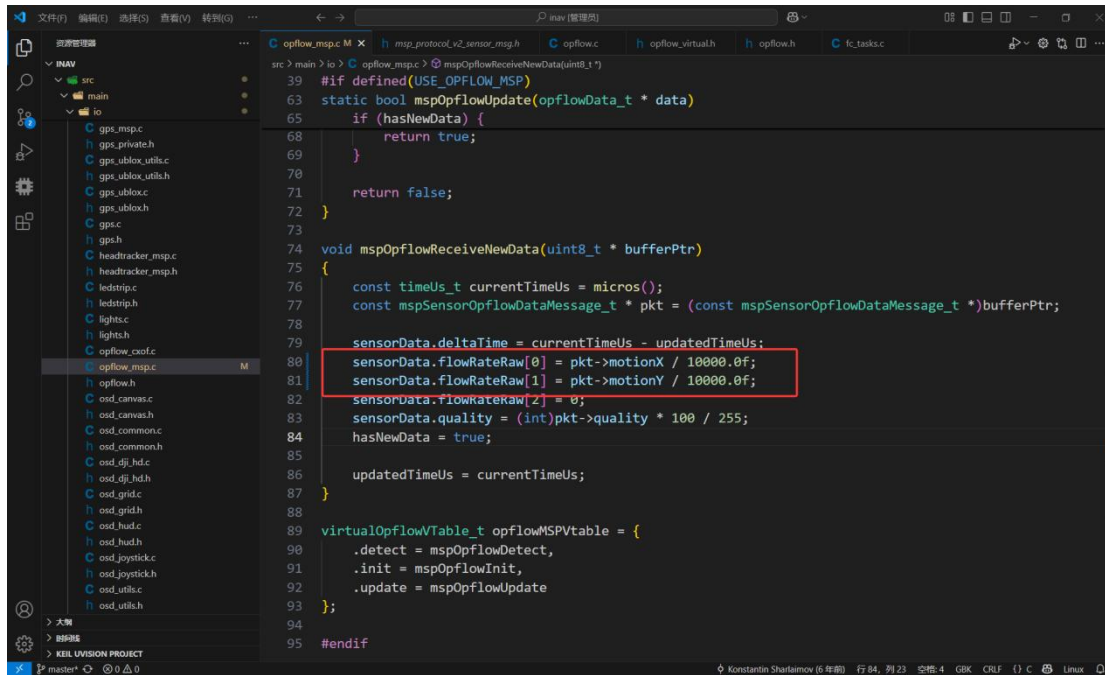
下载 INAV 固件后, 打开文件夹 `src/main/map/msp_protocol_v2_sensor_msg.h`, 可知 INAV 接收光流数据的类型为 `int32_t`, 而光流发送的数据是以 `真实值*10000` 的形式发送的, 故需要在接收到光流数据后做进一步处理。



打开文件夹 `src/main/io/opflow_msp.c` 在这里将接收到的数据除以 10000



修改后如图



最后保存即可

2 INAV 固件编译

2.1 环境安装

在编译固件之前先安装编译环境，在 linux 环境下：

① 安装交叉编译工具链：

```
sudo apt-get install gcc-aarch64-linux-gnu
```

```
sudo apt-get install g++-aarch64-linux-gnu
```

② 安装 gcc-arm-none-eabi：

```
sudo apt-get install gcc-arm-none-eabi
```

③ 安装 docker，并处理 docker 运行的权限问题：

```
sudo apt-get install docker.io
```

```
sudo chmod 666 /var/run/docker.sock
```

④ 安装依赖环境：

```
sudo apt install git make ruby cmake gcc
```

⑤ 更新软件包

```
sudo apt update
```

```
sudo apt upgrade
```

2.2 INAV 源代码的拉取和编译

在自己指定的文件夹下输入 `git clone https://github.com/INAVFlight/INAV.git`

```
niuma@Ubuntu: ~/github$ git clone https://github.com/INAVFlight/inav.git
```

输入 `cd INAV/` 进入 INAV 文件夹:

```
niuma@Ubuntu:~/github/inav$ ls
AUTHORS      build.sh      dev           downloads     lib           src
board        cmake         Dockerfile    fake_travis_build.sh  LICENSE       tools
build_docs.sh CMakeLists.txt docs          JLinkSettings.ini    readme.md     Vagrantfile
niuma@Ubuntu:~/github/inav$
```

输入 `git tag` 查看所有版本:

```
niuma@Ubuntu: ~/github/inav
5.0.0-RC1
5.0.0-RC2
5.0.0-RC3
5.1.0
6.0.0
6.0.0-FP1
6.0.0-RC1
6.0.0-RC2
6.0.0-RC3
6.1.0
6.1.0-RC1
6.1.1
7.0.0
7.0.0-RC1
7.0.0-RC2
7.0.0-RC3
7.1.0
7.1.0-RC1
7.1.1
7.1.2
8.0.0
8.0.0-RC1
8.0.0-RC2
8.0.0-RC3
8.0.0-RC4
8.0.1
8.0.1-RC1
8.0.1-RC2
INAV-1.2
v20240613.18
(END)
```

输入 `git checkout` 可以切换版本, 这里我们切换 8.0.1, 输入 `git checkout 8.0.1`

```
niuma@Ubuntu:~/github/inav$ git checkout 8.0.1
M      src/main/io/opflow_msp.c
HEAD 目前位于 ae47bcba0 Merge pull request #10718 from DusKing1/hugo-fix-SKYSTARSF405WING-target-config
niuma@Ubuntu:~/github/inav$
```

切换成功后, 可以使用 `git branch` 查看

```
niuma@Ubuntu:~/github/inav$ git branch
* (头指针分离于 8.0.1)
master
niuma@Ubuntu:~/github/inav$
```

在 INAV 文件下构建 build 目录，输入

```
mkdir build
```

```
cd build
```

```
cmake ..
```

```
niuma@Ubuntu:~/github/inav$ mkdir build
niuma@Ubuntu:~/github/inav$ cd build/
niuma@Ubuntu:~/github/inav/build$ cmake ..
```

此时 build 文件会多出一些文件

```
niuma@Ubuntu:~/github/inav/build$ ls
bin CMakeCache.txt CMakeFiles cmake_install.cmake Makefile src
```

在 build 文件夹下输入 `make xx(xx 为自己手中飞控板的型号)`，如

```
niuma@Ubuntu:~/github/inav/build$ make AOCODARCF7MINI_V1
```

当前手头硬件板子是否在 INAV 的支持列表中，可以在 `rsc/main/target` 文件夹中查看

```
niuma@Ubuntu:~/github/inav$ cd src/main/target/
niuma@Ubuntu:~/github/inav/src/main/target$ ls
AETH743Basic      DALRCF405          HAKRCKD722        MATEKF405SE       SPEDIXF405
AIKONF4           DALRCF722DUAL      HGLRCF405V2       MATEKF405TE       SPEDIXF722
AIKONF7           F4BY              HGLRCF722         MATEKF411         SPEEDYBEEF4
AIRBOTF4          FF_F35_LIGHTNING  IFLIGHT_2RAW_H743 MATEKF411SE       SPEEDYBEEF405AIO
AIRBOTF7          FF_FORTINIF4      IFLIGHT_BLITZ_ATF435 MATEKF411TE       SPEEDYBEEF405MINI
ALIENFLIGHTF4     FF_PIKOF4         IFLIGHT_BLITZ_F722 MATEKF722         SPEEDYBEEF405V3
ALIENFLIGHTNGF7   FIREWORKSV2       IFLIGHT_BLITZ_F722_X1 MATEKF722PX       SPEEDYBEEF405V4
ANVFC             FISHDRONEF4       IFLIGHT_BLITZ_F7_AIO MATEKF722SE       SPEEDYBEEF405WING
ANYFC7           FLASHHOBBYF405    IFLIGHT_BLITZ_F7_PRO MATEKF765         SPEEDYBEEF7
ANYFCM7          FLASHHOBBYF722    IFLIGHT_BLITZ_H7_PRO MATEKH743         SPEEDYBEEF745AIO
AOCODARCF405AIO   FLYW00F405PRO     IFLIGHTF4_SUCCXD    MICOAIR405MINI    SPEEDYBEEF7MINI
AOCODARCF4V2     FLYCOLORF7V2      IFLIGHTF4_TWING     MICOAIR405V2      SPEEDYBEEF7V2
AOCODARCF4V3     FLYW00F405PRO     IFLIGHTF7_TWING     MICOAIR743        SPEEDYBEEF7V3
AOCODARCF722AIO  FLYW00F405S_AIO   IFLIGHT_H743_AIO_V2 MICOAIR743AIO     SPRACING4EVO
AOCODARCF7DUAL   FLYW00F411        IFLIGHT_J8F7PRO     MICOAIR743V2      SPRACINGF7DUAL
AOCODARCF7MINI   FLYW00F722PRO     JHEF405PRO          NEUTRONRCF435MINI stm32f7xx_hal_conf.h
AOCODARCF7DUAL   FLYW00F745        JHEH7AIO            NEUTRONRCF435SE   stm32h7xx_hal_conf.h
ASGARD32F4       FLYW00F7DUAL      JHEMCUF405          NEUTRONRCF435WING system_at32f435_437.c
ASGARD32F7       FLYW00H743PRO     JHEMCUF405WING      NEUTRONRCH7BT     system_at32f435_437.h
ATOMRCF405NAVI   FOXEERF405        JHEMCUF722          OMNIBUSF4         system.h
ATOMRCF405NAVI_DELUX FOXEERF722DUAL    JHEMCUF745          OMNIBUSF7         system_stm32f4xx.c
ATOMRCF405V2     FOXEERF722V4      JHEMCUF743HD        OMNIBUSF7         system_stm32f4xx.h
AXISFLYINGF7PRO  FOXEERF745AIO     KAKUTEF4            OMNIBUSF7NXT      system_stm32f7xx.c
BEEROTORF4       FOXEERH743        KAKUTEF4WING        PIXRACER          system_stm32f7xx.h
```

编译完成后，显示如下

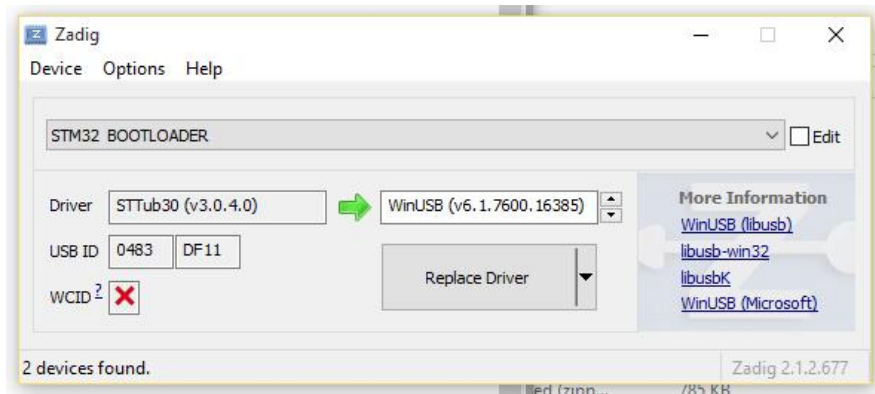
```
niuma@Ubuntu:~/github/inav/build
vlink.c.obj
Building C object src/main/target/AOCODARCF7MINI/CMakeFiles/AOCODARCF7MINI_V1.elf.dir/__/telemetry/ms
p_shared.c.obj
Building C object src/main/target/AOCODARCF7MINI/CMakeFiles/AOCODARCF7MINI_V1.elf.dir/__/telemetry/sb
us2.c.obj
Building C object src/main/target/AOCODARCF7MINI/CMakeFiles/AOCODARCF7MINI_V1.elf.dir/__/telemetry/sb
us2_sensors.c.obj
Building C object src/main/target/AOCODARCF7MINI/CMakeFiles/AOCODARCF7MINI_V1.elf.dir/__/telemetry/sm
artport.c.obj
Building C object src/main/target/AOCODARCF7MINI/CMakeFiles/AOCODARCF7MINI_V1.elf.dir/__/telemetry/si
m.c.obj
Building C object src/main/target/AOCODARCF7MINI/CMakeFiles/AOCODARCF7MINI_V1.elf.dir/__/telemetry/te
lemetry.c.obj
Linking C executable ../bin/AOCODARCF7MINI_V1.elf
lto-wrapper: warning: using serial compilation of 13 LTRANS jobs
lto-wrapper: note: see the '-flto' option documentation for more information
Memory region      Used Size  Region Size  %age Used
ITCM_RAM:          10568 B      16 KB      64.50%
ITCM_FLASH:         0 GB      16 KB      0.00%
ITCM_FLASH_CONFIG: 0 GB      16 KB      0.00%
ITCM_FLASH1:        0 GB      480 KB      0.00%
FLASH:             920 B      16 KB      5.62%
FLASH_CONFIG:       0 GB      16 KB      0.00%
FLASH1:            469149 B    480 KB     95.45%
TCM:                25192 B      64 KB     38.44%
RAM:               104360 B     192 KB     53.08%
MEMORY_B1:         0 GB      0 GB
Built target AOCODARCF7MINI_V1.elf
Scanning dependencies of target AOCODARCF7MINI_V1
Built target AOCODARCF7MINI_V1
niuma@Ubuntu:~/github/inav/build$
```

此时 build 文件夹中会多出一个 .hex 文件

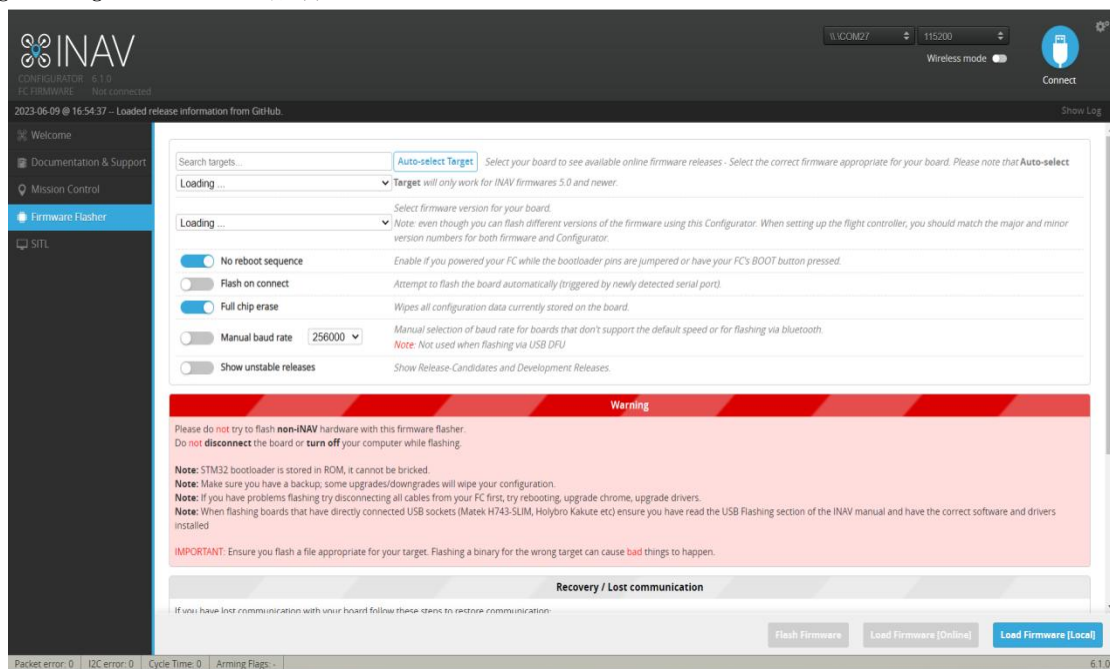
```
niuma@Ubuntu:~/github/inav/build$ ls
bin  CMakeCache.txt  CMakeFiles  cmake_install.cmake  inav_8.0.1_AOCODARCF7MINI_V1.hex  Makefile  src
```

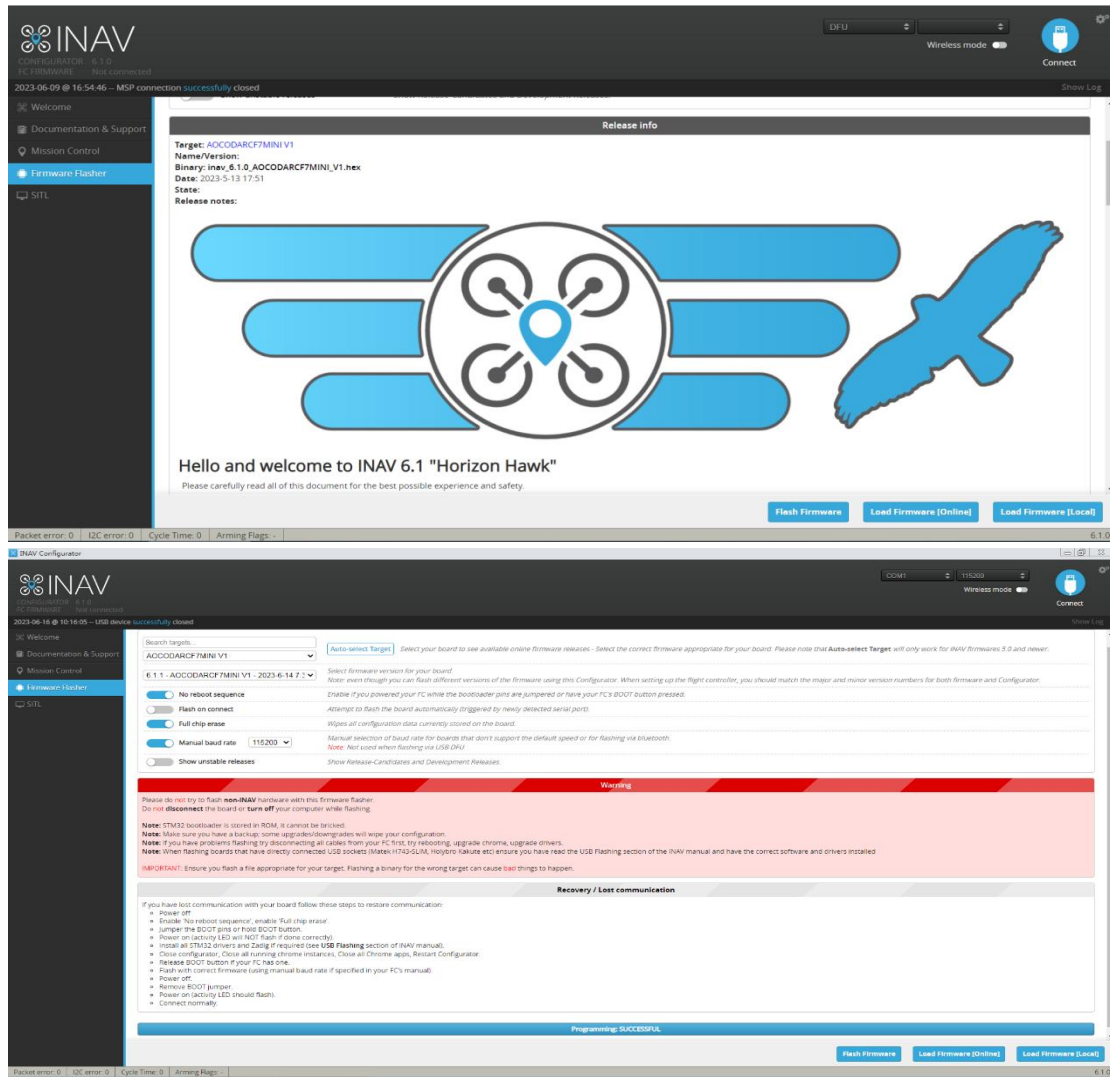
3 INAV 固件烧录

1、先按住板端按键再通过 USB 连接 PC, 此时进入 DFU 模式, 打开 Zadig 软件, 选择 STM32 BOOTLOADER, 选择 WinUSB, 点击 Replace Driver 安装驱动;



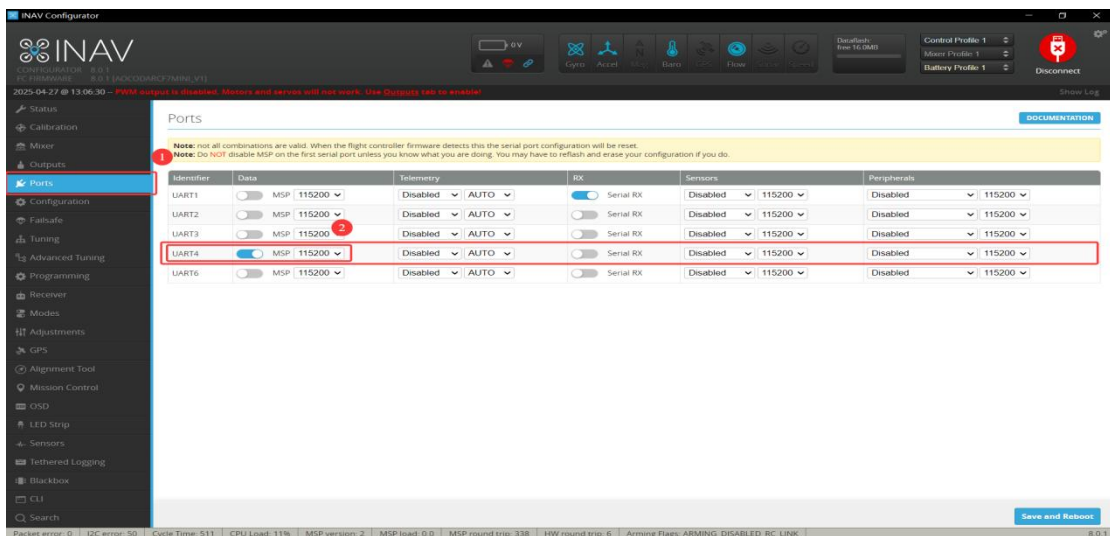
2、INAVConfigurator 烧录固件方法: 先按住板端按键再通过 USB 连接 PC, 此时进入 DFU 模式, 在主界面点击 Firmware Flasher 页面, 选择板子和固件型号, 勾选 No reboot sequence 和 Full chip erase, 选择 Load Firmware[Online]或 Load Firmware[Local]加载所需固件, 最后点击 Flash Firmware, 提示 Programming: SUCCESSFUL 成功。



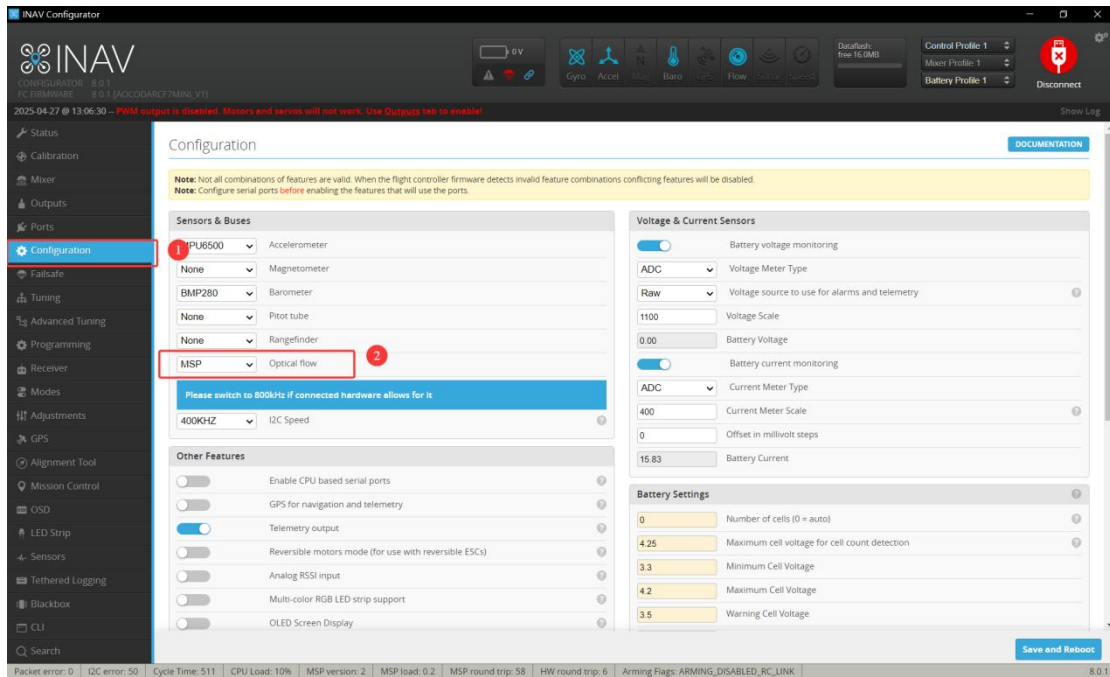


4 inva 地面站的使用

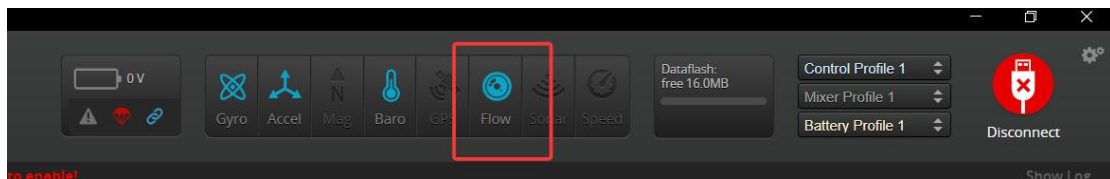
将飞控板连接上地面站后，将对应串口的 MSP 打开，设置波特率为 115200，保存并重启



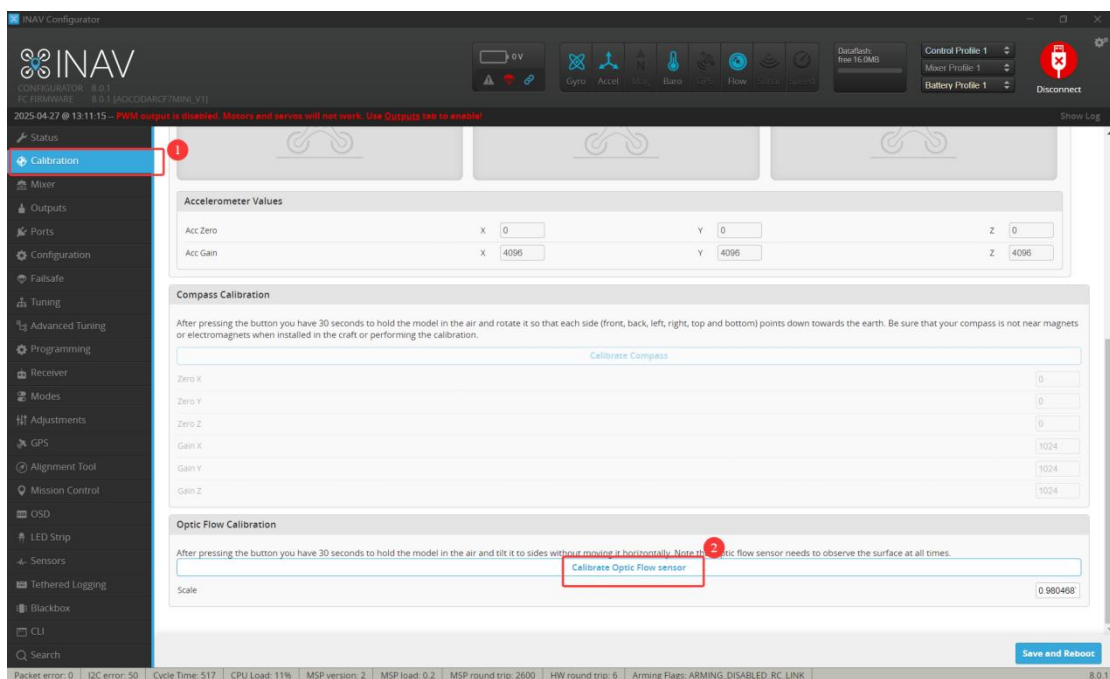
在传感器中，将测距仪和光流计的传感器都选择为 MSP，保存并重启。



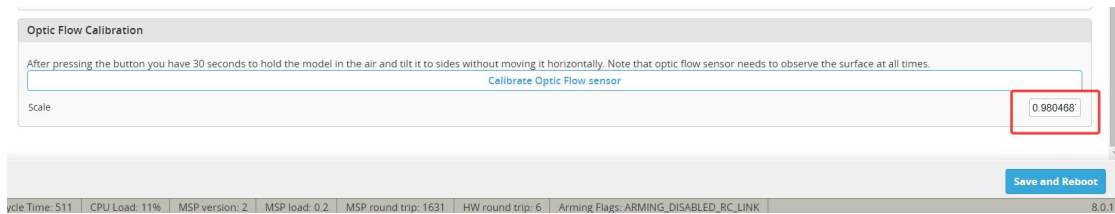
正常情况下，飞控应该已经识别了光流计，在顶部会有显示：



此时切换到校准页面，在右下角的模块中对光流计进行校准。这种校准是自动校准，如果自动校准的效果不好，也可以手动校准。



校准方法：将飞机轻轻拿起，在光流计下面铺一张有纹理的东西，比如一张印有文字的纸（报纸什么的都行），表面不要太反光就好。拿起飞机后，前后左右慢慢摆动飞机，不要让飞机发生移动就好，坚持 30 秒直到倒计时的对话框消失。你就会得到校准后的数据了。



官方建议使用以下 PID 以便光流计能更稳定的工作，于是需要在 CLI 输入以下命令：

```
set nav_mc_vel_z_p = 150
set nav_mc_vel_z_i = 250
set nav_mc_vel_z_d = 25
set nav_mc_pos_xy_p = 80
set nav_mc_vel_xy_p = 50
set nav_mc_vel_xy_i = 40
set nav_mc_vel_xy_d = 60
save
```

设置完成后，我们用传感器检查一下，看看是否正常。在传感器的调试模块，我们需要看光流计的数据，因此要先设置传感器显示光流计的数据，先输入以下命令：

```
set debug_mode = FLOW_RAW
save
```

接下来观察光流计的数据。

